CIIPS Glory
Soccer Robots with Local Intelligence

Thomas Bräunl
Dept. of Electrical and Electronic Engineering
The University of Western Australia
Nedlands, Perth, WA 6907
www.ee.uwa.edu.au/~braunl

Abstract. Our team was named after the new and successful *Perth Glory* soccer team. The heart of our robots are the *EyeBot* controllers, which we developed form scratch. We use a Motorola 68332 32-bit controller, which offers a variety of digital/analog I/O facilities. We developed our own operating system *RoBIOS* for these systems, which allows a great deal of flexibility. All image processing and planning is done locally on-board the *EyeBot*. We do not use any global sensor systems. The same *EyeBot* controller is also used for 6-legged and biped walking machines, and – as a boxed version – for undergraduate courses on assembly language programming.

1 Introduction

Our system architecture uses local intelligence without any global sensors. Although our approach is clearly disadvantaged with respect to performing well at a *RoboCup* competition, we do believe that our approach of truly autonomous and locally intelligent systems does make more sense for less restrictive applications. We are more interested in research on general purpose intelligent agents, as opposed to building a system which can serve only in a certain competition, has to rely on global sensors, and reduces mobile robots to remote controlled toy cars. We incorporated a digital color camera and a graphics display to our microcontroller system. All image processing is performed on-board.

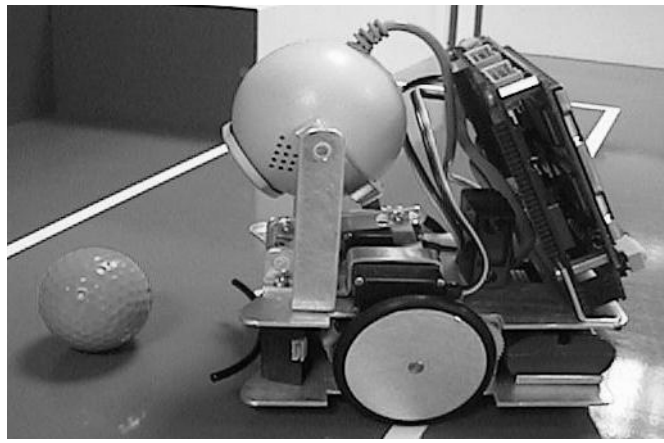

Fig 1. *EyeBot* soccer robot with local intelligence

M. Asada and H. Kitano (Eds.): RoboCup-98, LNAI 1604, pp. 416-421, 1999.

After our experience in the AAAI *Mobile Robot Competition* [2] with large industrial robots and equipment failure, we decided to develop completely new mini-robots to enter the small robot league of RoboCup. Each robot is driven by two DC motors in differential steering and is equipped with two shaft encoders, five infrared range sensors, and a digital color camera. Two PWM servos enable tilting of the camera and activation of the ball kicker bar. A wireless communication unit is currently being developed to allow the robots to talk to each other. We are able to operate without communication between the robots. Each robot is told its starting position and uses its shaft encoders plus infrared sensors to keep track of its current position. This limits our soccer strategies to finding the ball and heading for the goal. Although theoretically possible without robot communication, we will wait with more sophisticated behaviours like passing a ball to another robot, until wireless communication is in place. Since our processing unit is a relatively simple on-board microcontroller, there are obvious limitations on what can be done in terms of image processing in real time.

We are currently operating on images of size 60□80, which we found have sufficient resolution for detecting objects (ball, goals, players, walls) and navigating towards/ away from them. The controller is powerful enough to perform real-time on-line image processing, depending on the complexity of the operation.

Fig 2. *EyeBot MK3* controller

2 Hardware

Our extendable *EyeBot* base platform is a very compact board (about 9cm □□10 cm), which was developed around the key requirements of image processing. It therefore features a digital camera and an LCD graphics display. It is based on a Motorola 68332 microcontroller [4] and therefore also provides a sufficient number of I/O ports for the connection of various sensors and actuators or any future extensions. While most robot vision systems are either tethered or remote-controlled [1], on-board real-time vision is feasible for large and expensive mobile platforms. Although it seemed very difficult to implement real-time vision on a small and inexpensive system, *EyeBot* accomplishes this goal. *EyeBot* has been successfully used in the construction of a wheel-driven

vehicle, a 6-legged walking machine, and two biped walker robots. It is currently considered for the project of a flying robot.

The controller runs at a moderate speed (25 MHz), but it is fast enough to compute basic image operations on a low resolution image in real time. E.g. gray image acquisition, Sobel edge detection and display on the LCD for a 80 □ 60 image can be performed at a rate of more than 10 frames per second. *EyeBot*'s graphics LCD is essential for interaction between the robot and the programmer. One needs to see the robot's view in order to set camera parameters and orientation. Although the camera provides gray scale or color images at medium resolution, the display can only show low resolution black/white images. This is sufficient as a feedback to the programmer when running the robot, but not for program development, which is done on a workstation using the *Improv* tool.

While the hardware was started at Univ. Stuttgart, software design is an ongoing international joint research project between Univ. Stuttgart, Univ. Kaiserslautern (Germany), Rochester Institute of Technology (USA), and The Univ. of Western Australia.

3 Software

The operating system *RoBIOS* (**ro**bot **b**asic **i**nput **o**utput **s**ystem) has been implemented in C plus *m68k* assembly language, after adapting a version of the *gnu* C compiler and assembler tools for the *EyeBot*. This allows program development in a high level language, using assembly routines for time-critical passages [3]. *RoBIOS* comprises a small real time system with multi-tasking scheduler (essential for all robotics applications), libraries for various I/O functions (Table 1), and a number of demonstration applications. The microcontroller's timing processor unit (TPU) is being used for servo control with pulse width modulation (PWM), for sound synthesis and playback, as well as the control of infrared distance sensors.

KEY	key input, e.g. C routine getchar()
LCD	screen output, text and graphics
CAM	camera routines for grayscale and color
OS	operation system specific functions
MT	multi-tasking functions
SEM	semaphore operations
TIME	timer functions
RS232	serial line functions, e.g. prog. download
AUDIO	recording and playing back sounds
PSD	distance sensor routines
IR	infrared sensor readings
SONAR	sonar distance measurement
BUMP	acoustic collision detection
SERVO	servo positioning routines
MOTOR	servo motor control routines
LATCH	digital input/output and analog input
IMG	basic image processing routines

Table 1. *RoBIOS* function groups

The C low level text input and output routines have been adapted for *EyeBot*. This enables us to use the standard C I/O library *clib* together with the *EyeBot* system for user application programs. E.g., a user can call getchar(), in order to read a key input and use printf(..), in order to write text on the screen.

Special care has been taken to keep the *RoBIOS* operating system flexible among several different hardware configurations, because the same system is to be used for wheeled robots *and* for legged robots. Therefore, a hardware description table has been included into the system design, as described in the following. The *EyeBot* operating system *RoBIOS* relies on the **h**ardware **d**escription **t**able HDT, in order to find out which hardware components are currently connected to the system. These hardware components can be sensors or actuators (motors or servos), whose control routines are already available in the general *RoBIOS* system. HDT allows easy detection, initialization, and use of hardware components.

4 Tools

We developed a number of tools to facilitate robot programming. This is especially essential in the case of our mini-robots for which programs have to be cross-compiled and downloaded from a workstation.

4.1 Improv

Improv as a tool for designing the image processing part of a robot control program. It is an application running under Linux/X windows, using *EyeBot*'s digital camera. *Improv* displays the camera image at a higher resolution in a real time, together with five additional windows, representing user defined image processing stages. The *Improv* library comprises a number of low level image processing routines, while additional user defined operators can be added easily. The idea is to use *Improv* to design, test, and debug the robot's image processing component on a PC, until it has reached a stage where it can be tested on the vehicle. Then, the code needs to be recompiled for the microcontroller and downloaded to the robot.

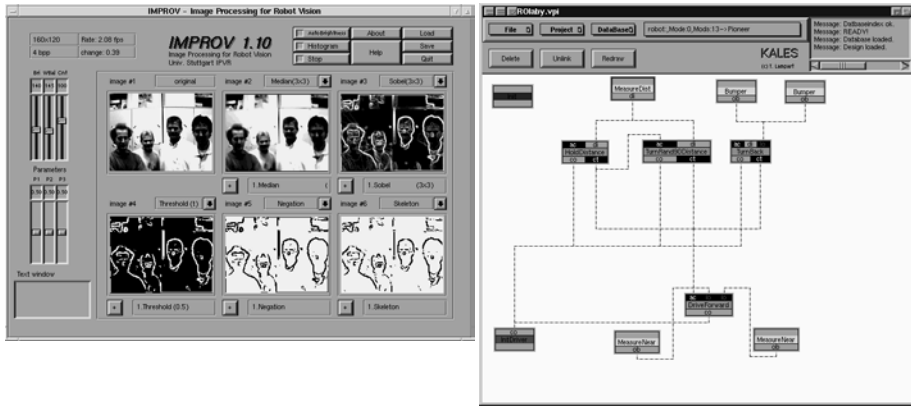

Fig 3. *Improv* real-time robot vision tool – *Rock&Roll* program development tool

4.2 Rock&Roll

On top of the operating system, we developed the integrated tool *Rock&Roll* (**ro**bot **c**onstruction **k**it and **ro**bot **l**ocomotion **l**ink) [5]. This system allows a "click-and-connect" construction of robot control structures. In its data flow model, sensors are sources and actuators are sinks, both representing system-defined module boxes. User-defined control boxes can be added, together with interconnection links between all modules, representing data flow.

4.3 EyeSim

The ***EyeBot*** **sim**ulator *EyeSim* is a valuable tool for the development of robot control programs. The simulator is actually implemented as a library with identical interface to the *RoBIOS* functions. That way a program can be compiled either way, for simulation or the real robot, without a change. Complex robot routines can be debugged and tested much more efficiently. Input/output routines can selected identical to the *EyeBot* LCD in a separate window or via Unix streams. *EyeSim* allows the concurrent simulation of multiple robots, several environment data formats, as well as the inclusion of moveable objects (here: soccer ball).

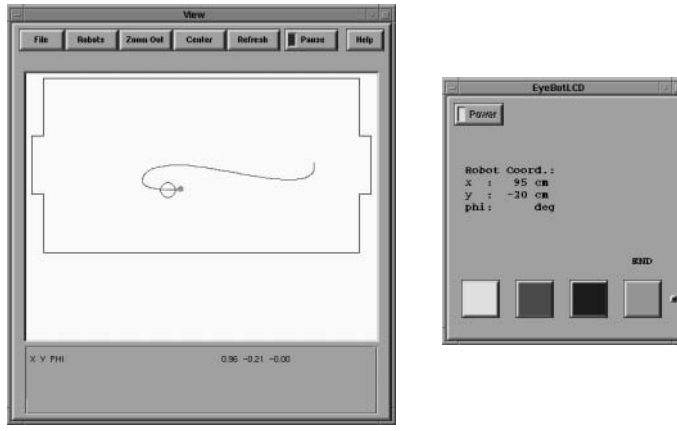

Fig 4. *EyeSim* mobile robot simulator

In Figure 4, the robot drives a spline curve towards the ball. The trajectory is generated by Hermite splines, involving the robot's start position and orientation, together with the perceived ball position and direction towards the goal. Intermediate control points are inserted in certain cases, e.g. when the robot is positioned between goal and ball.

5 Summary and Future Research

EyeBot robot developments include not only the robot soccer vehicles, but also several different 6-legged and biped walking machines. These are all using the same controllers and the same operating system, individually adapted the each robot's sensor/actuator configuration by the concept of a hardware description table.

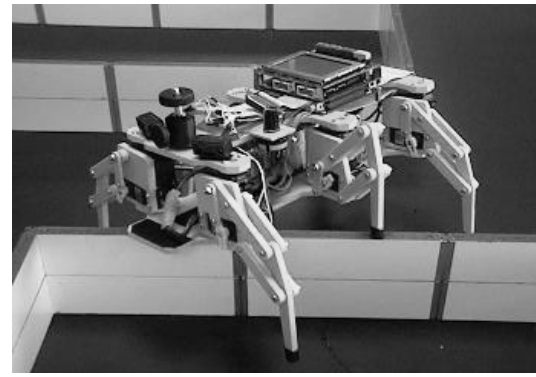

Fig 5. *EyeBot* walking robots are also members of the same family

We have discussed *EyeBot*, a platform used for mobile robots with local intelligence, allowing autonomous real-time vision control. Future research will concentrate on behaviour-based control models for groups of robot. More information is available on the Internet:

http://www.ee.uwa.edu.au/~braunl/eyebot/robots.html

Acknowledgments

The author acknowledges the work Jörg Henne (mechanics), Frank Sautter (electronics), Klaus Schmitt, Barbara Linn, Gerrit Heitsch, Michael Kasper (system software), Thomas Lampart (system software and programming tool), Nicholas Tay, Elliot Nicholls (simulator), and Birgit Graf (soccer software).

References

1. H. Bayer, Th. Bräunl, A. Rausch, M. Sommerau, P. Levi, *Autonomous Vehicle Control by Remote Computer Systems*, Proceedings of the 4th International Conference on Intelligent Autonomous Systems, IAS–4, Karlsruhe, March 1995, pp. 158–165 (8)
2. Th. Bräunl, M. Kalbacher, P. Levi, G. Mamier, *CoMRoS: Cooperative Mobile Robots* Stuttgart, Proc. 13. Nat. Conf. on Artificial Intelligence, AAAI Press, Portland OR, August 1996
3. The GNU Project, *GNU Documentation*, online, Delorie Software, www.delorie.com/gnu/docs/
4. Th. Harman, *The Motorola MC68332 Microcontroller*, Prentice Hall, 1991
5. P. Levi, M. Muscholl, Th. Bräunl, *Cooperative Mobile Robots Stuttgart: Architecture and Tasks*, Proceedings of the 4th International Conference on Intelligent Autonomous Systems, IAS–4, Karlsruhe, March 1995, pp. 310–317 (8)